Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Engineering Applications of Artificial Intelligence 16 (2003) 65–73
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

Extracting intuitive strokes in complex structured patterns with domain knowledge

Sung-Bae Cho*, Dong-Hyeop Han

Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, South Korea

Abstract

A standard iterative thinning algorithm which has been widely used to extract features for character recognition may destroy information due to several defects such as spurious loops in the skeleton and deformation in touched strokes. This is because most thinning algorithms rely on the steady erosion of character boundaries while maintaining the connectivity of the shape. To solve this problem, this paper proposes a knowledge-based thinning method which removes the spurious loops by a preprocessing stage and makes use of average stroke widths and domain knowledge on Hangul (Korean script) to extract intuitive strokes. The experimental results on 2000 handwritten Hangul characters in PE92 benchmark database indicate that the proposed method has reduced the number of defects and led to more intuitive strokes.

Keywords: Extraction of intuitive strokes; Knowledge-based thinning; Spurious loops; Line distortions; Offline handwritten characters; Hangul (Korean script)

1. Introduction

Thinning is a very important preprocessing step for the structural pattern analysis and recognition of various types of images. This reduces the width of a line pattern to just one single pixel, while preserving the feature of original pattern. Due to this merit, thinning has been widely used in many recognition problems such as character, fingerprint, chromosome, and so on (Hilditch, 1968).

Many thinning algorithms have been suggested and one of them, which has been widely used, is to reduce pattern boundaries repeatedly until it becomes to be skeleton. These iterative algorithms are classified as sequential or parallel depending on the criteria used for pixel deletion (Lam et al., 1992). Sequential thinning method makes use of both the results from the previous and current steps to decide whether the pixel should be deleted or not. On the other hand, parallel thinning method uses only the result from the previous step. The thinning result depends on the characteristics of the algorithms used.

Whatever the algorithm is adopted, however, thinning may cause distortions and faults in the result image due to the basic nature of the iterative algorithm without considering the information and characteristics of whole pattern (Lam and Suen, 1995). Fig. 1 shows a couple of examples for spurious loops between two strokes, and Fig. 2 shows a line distortion at touched strokes. This paper proposes a preprocessing method with templates to eliminate the pixels which cause the spurious loops, and a knowledge-based thinning method with the information of average stroke width and structure of Hangul (Korean script) to extract intuitive strokes.

The proposed method takes the following steps to remove the distortions. The pixel producing spurious loops like in Fig. 1 is eliminated by several templates in preprocessing step. The preprocessed image is thinned as many times as the number of average stroke width, and the touched strokes are separated by using the information of Hangul characters. Chen's algorithm (Chen and Hsu, 1988) is used as the basic thinning algorithm in this paper, because it prevents from serious shrinking and preserves 8-neighbor connectivity. In order to show the effectiveness of our knowledge-based approach, we have used 25 Hangul character files composed of 80 characters which are randomly extracted from PE92

*Corresponding author. Tel.: +82-2-2123-2720; fax: +82-2-2123-2579.
E-mail address: sbcho@csai.yonsei.ac.kr (S.-B. Cho).

0952-1976/03/$ - see front matter
doi:10.1016/S0952-1976(03)00025-3

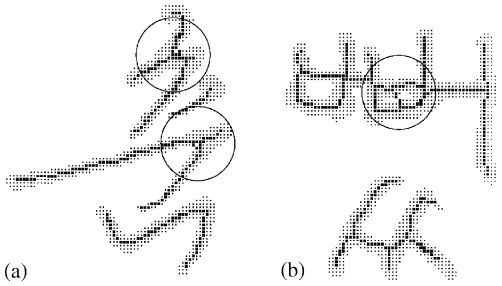


Fig. 1. Spurious loops between two strokes.

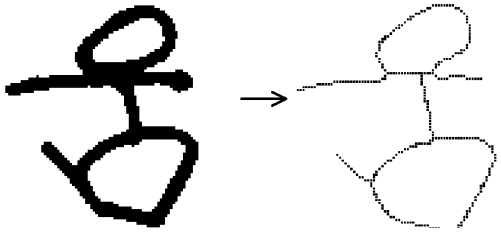


Fig. 2. Line distortion at touched strokes.

benchmark database (Kim et al., 1996). The total number of characters is amount to 2000.

This paper is organized as follows. We introduce the related works on the knowledge-based thinning and the distortions generated by conventional thinning algorithms with PE92 in Section 2. The proposed knowledge-based thinning algorithm is described in Section 3. In Section 4, experimental results are given.

2. Backgrounds

2.1. Structure of Hangul characters

The Korean script system Hangul consists of 24 characters each of which represents a phoneme. Ten of these are vowels and the rest are consonants. Characters are grouped together to form syllables. A syllable may consist of two to six characters. More than 11,000 syllables exist, but about 3000 suffice for ordinary use. A word consists of a sequence of syllables.

The rules of character combination for making a syllable at first seem complicated but logical. The Korean vowels are shaped either vertically or horizontally elongated. The vertical vowels have their accompanying consonants on their left and the horizontal vowels have their accompanying consonants on their top. If a syllable has a consonant after a vowel, it is always written below the main vowel. Depending on its position and accompanying character, the shape of a character varies.

The general structure of Hangul syllables is presented in Fig. 3, where V1 indicates a vertically shaped vowel, V2 a horizontally shaped vowel, C1 a head consonant,

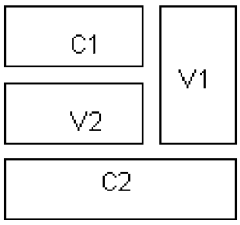


Fig. 3. General structure of Hangul syllables. V1 signifies a vertically shaped vowel, V2 a horizontally shaped vowel, C1 a head consonant, and C2 a bottom consonant.

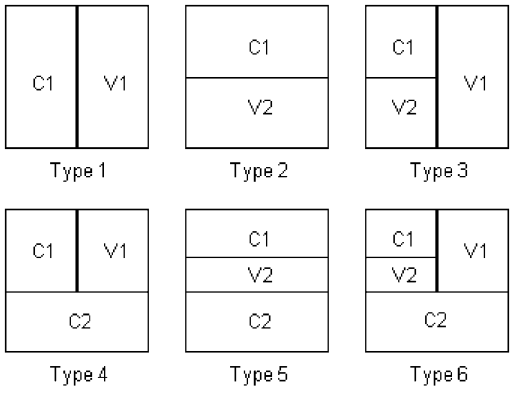


Fig. 4. Six types of Hangul syllables. They are based on the shape of the vowel included in the syllable, and on the presence or absence of a bottom consonant.

and C2 a bottom consonant. According to the shape of the vowel included in the syllable and the presence or absence of a bottom consonant, Hangul syllables can be divided into six categories, as shown in Fig. 4.

2.2. Types of distortion

Distortions caused by a thinning algorithm can be classified into three types: spurious loops between the two strokes (102 instances in our test database), line distortion at touched strokes (40 instances in our test database), and Y-shape distortion at crossing regions (38 instances in our test database). The spurious loops are simple but most frequently occurred. The line distortion at touched strokes is very difficult to be solved, because it depends on the domain knowledge. In this paper, we have tried to solve these two distortions.

2.3. Related works

Many knowledge-based thinning algorithms have been proposed so far. Stentiford proposes several templates used in preprocessing step (Stentiford and Mortimer, 1983). Specific defects in data can cause thinning algorithms to destroy information, leading to misrecognition. Therefore, this defect may produce topologically incorrect skeletons with spurious loops as shown in Fig. 1. The templates are aimed at deleting

pixels which cause spurious loops. In preprocessing step, all templates scan data and extract pixels which cause spurious loops using the knowledge of the structure of alphanumeric characters. Stentiford eliminates those pixels before thinning process and it can prevent spurious loops. However, it cannot solve all spurious loops because of the lack of knowledge of the structure of characters.

KBTA (Li and Suen, 1991) proposed by Suen is to apply shape knowledge to thin the crossing regions and merge these results with those obtained by another algorithm to thin the remaining regions. Compared with several thinning algorithms, it has better results at crossing regions, but takes longer processing time to get shape knowledge. Tseng extracts strokes directly from the original pattern (Tseng and Chung, 1992). It is based on the knowledge of the structure of Chinese characters, which extracts strokes effectively without iterative thinning process. This sort of non-iterative approach might be efficient when there are many complicated strokes in the patterns like Chinese characters.

Suzuki proposes the cross section sequence graph which describes line images in a simple and well structured form (Suzuki and Mori, 1993). This algorithm solves the deformation at crossing regions using the knowledge of boundary shape. It operates fast and effectively, but it requires much memory space. Finally, Suzuki proposes another graph-based method that suppresses shape distortions at crossing point with some predefined rules (Suzuki and Ueda, 1993). It may take much processing time since it uses the thinned result to remove the distortions at crossing regions.

As stated above, most knowledge-based thinning algorithms deal with the distortions generally occurred at crossing regions. In practical images, however, distortions may occur at various regions such as the one in touched strokes. The two strokes at this region become one stroke if conventional knowledge-based thinning algorithms are used. To solve this problem, we propose a new method which makes use of the domain knowledge.

3. Knowledge-based thinning algorithm

The basic thinning algorithm used in this paper is Chen's that is a parallel thinning algorithm (Chen and Hsu, 1988). This algorithm produces better contour noise immunity than sequential thinning algorithms (Naccache and Shingal, 1984), and solves both the 4-connectivity problem and the serious line shrinking happened in Zhang and Suen's algorithm (Zhang and Suen, 1984). This section describes the knowledge-based thinning algorithm in detail (Fig. 5).

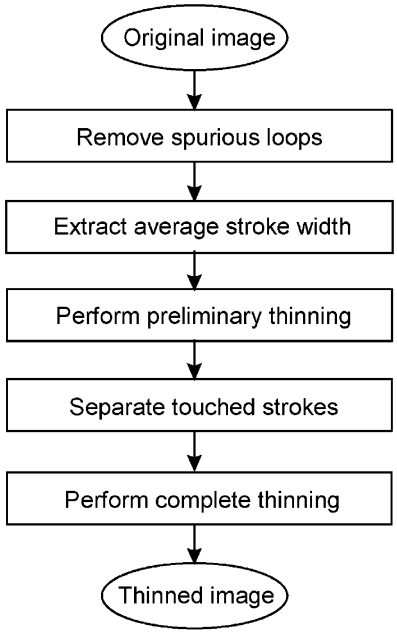


Fig. 5. Proposed algorithm.

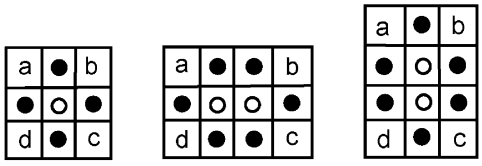


Fig. 6. Templates with white pixel in the center.

3.1. Removal of spurious loops

The pixel which causes spurious loops can be eliminated in preprocessing step with 3×3 or 3×4 templates. The templates can be classified into two types: one based on white pixel and the other based on black pixel.

3.1.1. Templates based on white pixel

Fig. 1(a) shows a couple of spurious loops occurred when white pixel is surrounded with black pixels in original image. These black pixels remain as spurious loop after thinning, because Chen's algorithm tends to preserve connectivity between black pixels. To prevent the spurious loops, these black pixels should be eliminated or white pixels surrounded with the black pixels should be replaced with black pixels. We implement this algorithm by applying the templates shown in Fig. 6 to the image in preprocessing step. If all pixels a, b, c and d in Fig. 6 are black pixels, the white pixels surrounded with black pixels are replaced with black pixels. If "pixel a" is white pixel, two black pixels adjacent to it are replaced with white pixels. Similarly,

the cases of "pixel b," "pixel c" and "pixel d" can be processed.

3.1.2. Templates based on black pixel

The spurious loop in Fig. 1(b) has nothing to do with white pixels. This distortion can be removed by templates in Fig. 7. If "pixel a" is black pixel, the black pixel in the center may remain to form spurious loops after thinning. Therefore, to prevent spurious loops, black pixel in the center should be eliminated before thinning process.

3.1.3. Application of templates

Fig. 8 shows the thinned result after applying the two templates to PE92 data before thinning. The spurious loops in '춤' character of the first image have occurred between very near two strokes. The angle between these two strokes is very acute. Templates in Fig. 6 can prevent this distortion. Two spurious loops in '춤' can be detected by the first template in Fig. 6. Upper spurious loop happens when "pixel b" is white pixel. In this case, "pixel a," "pixel c" and "pixel d" are all black pixels. Two black pixels adjacent to "pixel b" is eliminated to prevent the spurious loop. Lower spurious loop happens when "pixel d" is white pixel. It is easy to find which pixel should be eliminated from preventing upper spurious loop. On the other hand, the spurious loop such as in '뻤' character of the second image has occurred when two boundary pixels are popped out and connected. The black pixels caused the spurious loop can be deleted by the third template in Fig. 7.

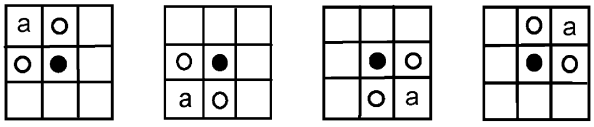


Fig. 7. Templates with black pixel in the center.

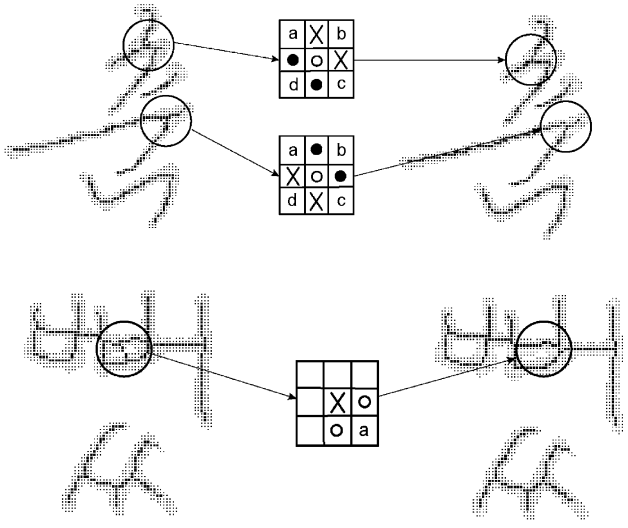


Fig. 8. Results of applying the templates to prevent spurious loops.

3.2. Preliminary thinning

To extract intuitive strokes by separating touched strokes, we have to know the average stroke width in a character image. It is easily obtained by calculating the run lengths of all line segments horizontally and vertically. The highest frequency of run lengths is decided as average stroke width. Then, preliminary thinning has been performed until the number of thinning operation is up to the average stroke width. The basic thinning is conducted by Chen's algorithm which consists of two subcycles. First subcycle decides whether the boundary pixels on the right or top of the stroke should be deleted or not, and eliminates the boundary pixel while preserving the connectivity of skeleton. Second subcycle does the same work at pixels on the left or bottom of the stroke. Thinning is to reduce the width of a line pattern to just one single pixel. Therefore, if two strokes are touched in original image, the line distortion might frequently happen as shown in Fig. 2.

The preliminary thinning ends when the number of thinning operation reaches to the average stroke width. Therefore, the preliminary thinning does not lose the information of touched region. Fig. 9 shows the result of the image in Fig. 2 at this stage. In the case of Fig. 2, the average stroke width is four, and thereby the preliminary thinning performs by four iterations. After this process, any touched region may remain as more than one pixel width. To obtain correct result in touched regions, it is important to extract and separate them appropriately.

3.3. Separation of touched strokes

Touched region produces a line distortion. The proposed method extracts candidate touched regions and decides the type of them with the structural information of Hangul. Appropriate rules are applied according to the type of touched regions.

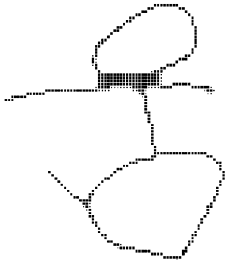


Fig. 9. The result of preliminary thinning for the number of the average stroke width.

3.3.1. Extraction of touched region

After the preliminary thinning is finished, the extraction process of touched region begins as follows. T is defined as half the number of average stroke width.

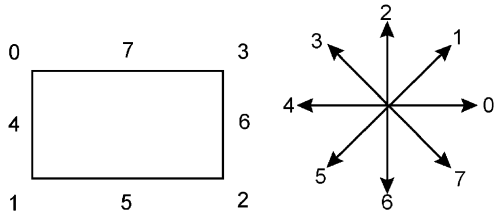
- *Step* 1: Check whether horizontal line segment longer than T exists or not.
- *Step* 2: Extract touched region whose vertical length is longer than T.
- *Step* 3: Decide the shape of touched region. Calculate the number and direction of strokes connected to it.
- *Step* 4: Divide the touched regions by the rules such as making hole, divide and dig, according to knowledge of the structure of Hangul, the characteristics of stroke and the information obtained from Step 3.
- *Step* 5: Repeat from Step 1 to Step 4 until there remains no more touched region in the image.

The shape of touched region in Step 3 is one of three types: square, horizontal rectangle and vertical rectangle. The different types of shape might be more than three, but three types are enough to cover all various shapes in our problem. According to the information of touched regions and the number and directions of strokes connected to touched regions, one of three rules is applied to the image.

3.3.2. Representation of stroke information

After deciding the shape of touched region, we check the strokes which are connected to touched region. If a connected stroke to the touched region is longer than threshold, this stroke is decided to be connected with the touched region. The threshold is set as four because average stroke width is usually four in many characters. Each stroke is represented as a pair (position code:direction code), shown in Fig. 10.

Therefore, each touched region is identified by the shape, the number of connected strokes, and a set of (position code:direction code) of connected strokes. For example, the touched region in Fig. 11 has the information as follows. The shape of touched region is horizontal rectangle, the number of connected strokes is five, and the set of (position code:direction code) are (0:2), (1:4), (2:0), (3:1) and (5:6).



(a) position code (b) direction code

Fig. 10. Position and direction codes of a stroke.

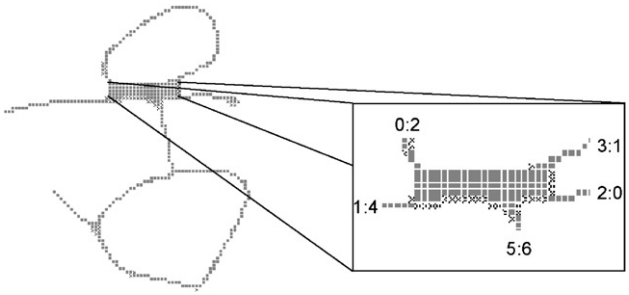


Fig. 11. An example of the information extracted from touched region.

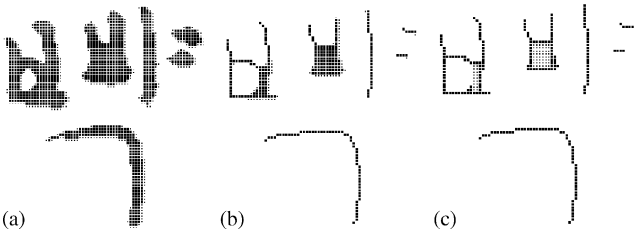


Fig. 12. An example of applying the type 1 rule: Making hole.

3.3.3. Types of rules

The rules to separate touched regions are of one of three types: making hole, horizontal/vertical divide and horizontal/vertical dig. After these rules are applied to the image, the remaining image is thinned by Chen's algorithm to one pixel wide. Conditions for the rules do not have to cover all possible combinations because they do with the distortions that might occur in real images. However, the conditions are devised not to be overlapped by considering the construction rule of Hangul and the basic information about distortions. Also, new rules can be easily added.

(1) Type 1: *Making hole*. The rule of making hole is for consonants such as 'ㅇ' and 'ㅂ'. Fig. 12(a) shows a distorted consonant 'ㅂ' in '빡' character, (b) shows the result of the preliminary thinning, and (c) shows the result after applying the making hole rule with information extracted from touched region. Table 1 shows the conditions applied to making hole rules. In this table, when the number of connected strokes is zero or one, it means consonant 'ㅇ' without hole. If the number of connected strokes is two and the positions of connected strokes are up-left (0:2) and up-right (3:2), it means consonant 'ㅂ' without hole.

(2) Type 2: *Divide*. Divide rules of touched region are for the vowels such as 'ㅡ', 'ㅜ', 'ㅗ' and every consonant adjacent to them. Fig. 13 shows (a) the original image, (b) the result of the preliminary thinning, and (c) the result after divide rule is applied, respectively. Table 2 shows the conditions applied to the divide rules. Horizontal divide rule is for the touched region composed of two horizontal strokes and vertical divide rule is for the touched region of two vertical strokes.

Table 1
Rules of making hole

	Conditions			Rule
	Number of connected strokes	Position code: direction code	Shape of touched region	
1	0	Don't care	Don't care	Making hole
2	1			
3	2	0:2, 3:2	Square	

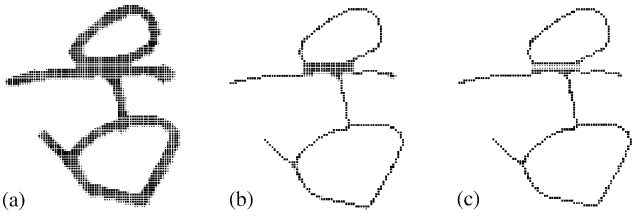


Fig. 13. An example of applying the type 2 rule: Horizontal divide.

Table 2
Rules of divide

	Conditions			Rule
	Number of connected strokes	Position code: direction code	Shape of touched region	
1	2	0:4, 3:0	Horizontal rectangle	Horizontal divide
2	2	1:4, 2:0		
3	3	0:4, 3:0		
4	3	1:4, 2:0		
5	4	0:4, 3:0		
6	4	1:4, 2:0		
7	More than 5	0:4, 3:0		
8	More than 5	1:4, 2:0		
9	2	0:2, 3:2	Vertical rectangle	Vertical divide
10	2	1:6, 2:6		
11	3	0:2, 3:2		
12	3	1:6, 2:6		
13	4	0:2, 3:2		
14	4	1:6, 2:6		
15	More than 5	0:2, 3:2		
16	More than 5	1:6, 2:6		

(3) Type 3: *Dig*. Like the divide rules, dig rule applies to the case of two parallel touched strokes connected with another stroke perpendicularly. This case occurs at vowels such as 'ㅖ', 'ㅑ', 'ㅛ' and 'ㅠ', and consonants such as 'ㅌ', 'ㅍ' and 'ㅋ'. Table 3 shows the conditions applied to the dig rules, and Fig. 14 shows the result after the dig rule is applied to.

Leftward dig rule is for touched region which occurs in vowel 'ㅑ'. Rightward dig rule is for touched region

Table 3
Rules of dig

	Conditions			Rule
	Number of connected strokes	Position code: direction code	Shape of touched region	
1	2	0:2, 1:6	Horizontal rectangle	Leftward dig
2	3	0:2, 1:6, 2:0		
3	3	0:2, 1:6, 3:0		
4	4	0:2, 1:6, 2:0, 3:0		
5	2	2:6, 3:2		Rightward dig
6	3	2:6, 3:2, 0:4		
7	3	2:6, 3:2, 1:4		
8	4	0:2, 1:6, 2:0, 3:0		
9	2	1:4, 2:0	Vertical rectangle	Downward dig
10	3	1:4, 2:0, 0:2		
11	3	1:4, 2:0, 3:2		
12	4	1:4, 2:0, 0:2, 3:2		
13	2	0:4, 3:0		Upward dig
14	3	0:4, 3:0, 1:6		
15	3	0:4, 3:0, 2:6		
16	4	0:4, 3:0, 1:6, 2:6		

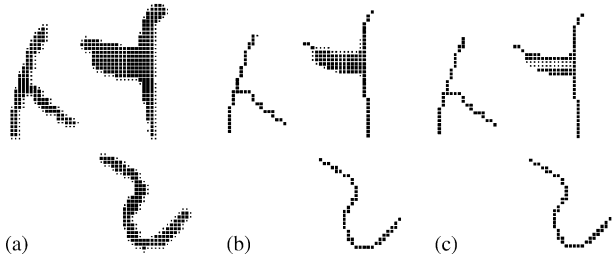


Fig. 14. An example of applying the type 3 rule: Rightward dig.

which occurs in vowel 'ㅖ'. Downward and upward dig rules are for 'ㅛ' and 'ㅠ', respectively.

3.3.4. Implementation of rules

The rules are implemented based on the knowledge of the structure of Hangul, the characteristics of stroke and the information of connected strokes. For a touched region extracted, a search is performed on a rule application tree in Fig. 15. T is defined as half the number of average stroke width. NS is the number of strokes connected to touched region. Stroke information is represented as (position code:direction code).

For an example, in Fig. 16 the shape of touched region is decided as horizontal rectangle, because horizontal length of the touched region is longer than T, say 4. It indicates that one of the rules of horizontal divide, rightward dig and leftward dig might be applied. With some empirical studies, we find that NS is 2. Since the information of connected strokes is (3:2) and (2:6), touched region can be considered as a part of vowel 'ㅖ', so that rightward dig has to be applied.

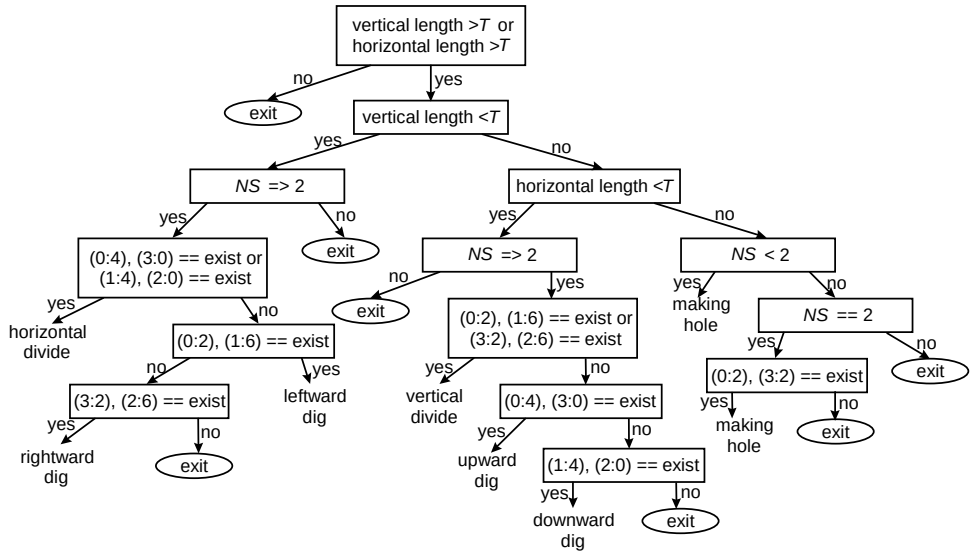


Fig. 15. A diagram of the rule application tree. T is half the number of average stroke width, and NS is the number of strokes connected to touched region.

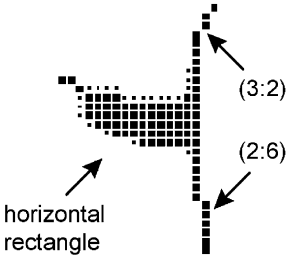


Fig. 16. Touched region as a part of vowel 'ㅕ'.

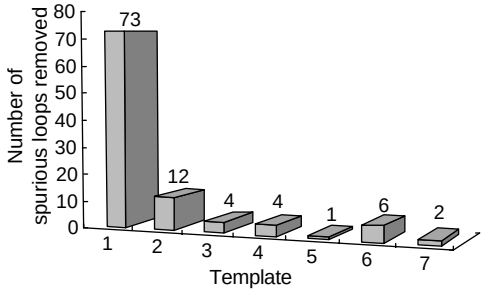


Fig. 17. The numbers of spurious loops eliminated by each template.

4. Experimental results

In order to show the usefulness of the proposed method, we have conducted the experiments with PE92 benchmark database. The proposed algorithm is implemented with Visual C++ on 160 MHz Pentium PC. The computation time is about 0.297 s per character. It depends on the size of data and writing device. The test images are normalized to 100×100 grids.

4.1. Removal of spurious loops

Fig. 17 shows the distribution of the spurious loops eliminated with respect to each template. As it can be seen, all the 102 spurious loops occurred in 2000 test data are removed by seven templates proposed in Figs. 6 and 7. It can be seen that most of the spurious loops are solved by the first template in Fig. 6.

4.2. Separation of touched strokes

The proposed algorithm has solved 36 out of 40 line distortions. Fig. 18 shows the distribution of line distortions eliminated by each rule. Only four samples

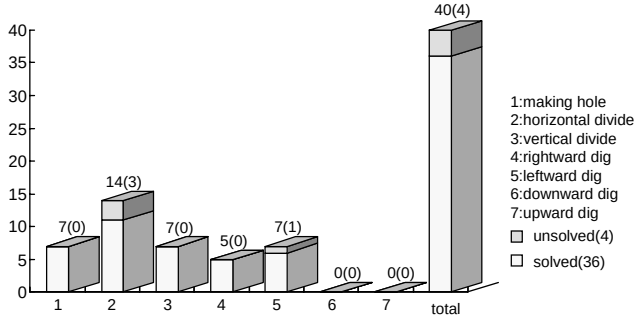


Fig. 18. The numbers of line distortions solved by each rule.

remain unsolved as shown in Fig. 19, where touched region of the data cannot be extracted because the width of touched stroke is smaller than the average stroke width. This algorithm causes new 10 errors in 2000 data, all of which have a stroke whose width is twice as thick as average stroke width. In this case, the proposed algorithm has confused the stroke as two touched strokes, and tried to separate it into two strokes. Additional processing step would be required to work out this problem.

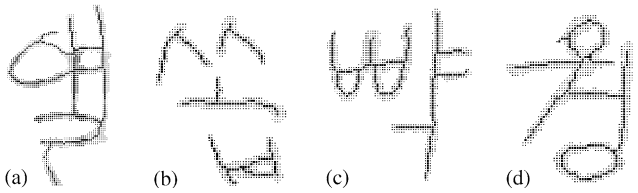
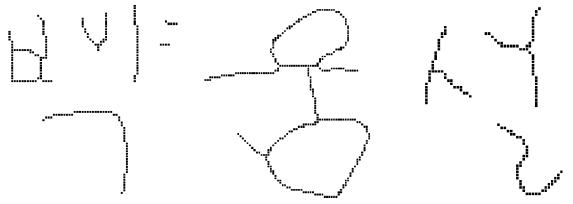
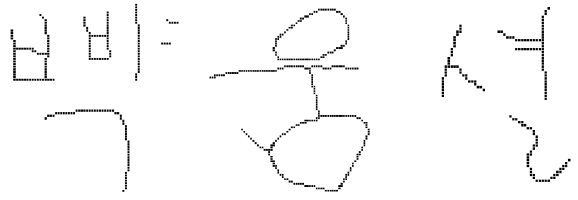


Fig. 19. Images unsolved by the proposed thinning method.

(a) Original images



(b) Thinned results by the conventional method



(c) Thinned results by the knowledge-based thinning method

Fig. 20. Comparisons of the thinning methods.

Fig. 20 compares the results obtained from basic thinning and proposed knowledge-based thinning methods. This figure illustrates that the proposed thinning method has produced more correct and intuitive results.

5. Concluding remarks

In this paper, we have proposed a knowledge-based thinning method which prevents spurious loops by preprocessing and one-line distortions between touched strokes by two-step thinning with knowledge base. It is not easy for conventional thinning methods to prevent spurious loops, and nearly impossible to solve one-line distortion. Most of them have been solved by the knowledge-based thinning method proposed in this paper. With the result of experiments on PE92 data, we can assert that the proposed algorithm is very useful and efficient.

In Section 2, we have introduced several conventional knowledge-based thinning algorithms. Both KBTA and Suzuki's method have tried to solve the deformation at crossing region by using the knowledge about data. Whereas all the conventional methods cannot solve the one-line distortion at touched regions, the proposed method solves it easily and efficiently, so that preserves the features of original images very well. The proposed approach to extracting intuitive strokes in Hangul characters is expected to be easily applied to other complex structured patterns like Chinese characters.

Acknowledgements

This work was supported in part by Biometrics Engineering Research Center, KOSEF, in Korea.

References

Chen, Y.S., Hsu, W.H., 1988. A modified fast parallel algorithm for thinning digital patterns. Pattern Recognition Letters 7, 99–106.

Hilditch, C.J., 1968. An application of graph theory in pattern recognition. Machine Intelligence 3, 325–347.

Kim, D.-H., et al., 1996. Handwritten Korean character image database PE92. IEICE Transactions on Information and Systems E79-D (7), 943–950.

Lam, L., Lee, S.W., Suen, C.Y., 1992. Thinning methodologies—a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (9), 869–885.

Lam, L., Suen, C.Y., 1995. An evaluation of parallel thinning algorithms for character recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17 (9), 914–919.

Li, B., Suen, C.Y., 1991. A knowledge-based thinning algorithm. Pattern Recognition 24 (12), 1211–1221.

Naccache, N.J., Shingal, R., 1984. SPTA: a proposed algorithm for thinning binary patterns. IEEE Transactions on Systems Man and Cybernetics 14 (3), 409–418.

Stentiford, F.W.M., Mortimer, R.G., 1983. Some new heuristics for thinning binary handprinted characters for OCR. IEEE Transactions on Systems Man and Cybernetics 13 (1), 81–84.

Suzuki, T., Mori, S., 1993. Structural description of line images by the cross section sequence graph. International Journal of Pattern Recognition and Artificial Intelligence 7 (5), 1055–1076.

Suzuki, S., Ueda, N., 1993. Graph-based thinning for binary images. International Journal of Pattern Recognition and Artificial Intelligence 7 (5), 1009–1030.

Tseng, L.Y., Chung, C.T., 1992. An efficient knowledge-based stroke extraction method for multi-font Chinese characters. Pattern Recognition 25 (12), 1445–1458.

Zhang, T.Y., Suen, C.Y., 1984. A fast parallel algorithm for thinning digital patterns. Communications of the ACM 27 (3), 236–239.

Sung-Bae Cho received the B.S. degree in computer science from Yonsei University, Seoul, Korea, in 1988 and the M.S. and Ph.D. degrees in computer science from KAIST (Korea Advanced Institute of Science and Technology), Taejeon, Korea, in 1990 and 1993, respectively. He worked as a Member of the Research Staff at the Center for Artificial Intelligence Research at KAIST from 1991 to

1993. He was an Invited Researcher of Human Information Processing Research Laboratories at ATR (Advanced Telecommunications Research) Institute, Kyoto, Japan from 1993 to 1995, and a Visiting Scholar at University of New South Wales, Canberra, Australia in 1998. Since 1995 he has been an Associate Professor in the Department of Computer Science, Yonsei University. Dr. Cho was awarded outstanding paper prizes from the IEEE Korea Section in 1989 and 1992, and another one from the Korea Information Science Society in 1990. He was also the recipient of the Richard E. Merwin prize from the IEEE Computer Society in 1993. He was listed in Who's Who in Pattern Recognition from the International Association for Pattern Recognition in 1994, and received the best paper awards at International Conference on Soft Computing in 1996 and 1998. Also, he received the best paper award at World Automation Congress in 1998, and listed in Marquis Who's Who in Science and Engineering in 2000 and in Marquis Who's Who in the World in 2001. He is a Member of the Korea Information Science Society, INNS, the IEEE Computer Society, and the IEEE Systems, Man, and Cybernetics Society. His research interests include neural networks, pattern recognition, intelligent man-machine interfaces, evolutionary computation, and artificial life.

Dong-Hyeop Han received his B.S. and M.S. degrees from Yonsei University, Seoul, Korea, in 1996 and 1998, respectively. From 1996 to 1997, he joined the project supported by the consortium of offline handwritten character recognition in Korea. Since 1998 he has been appointed as a researcher in Samsung Data Communication Company. His fields of interest include pattern recognition, artificial intelligence, knowledge-based thinning and data mining.